Universidad ORT Uruguay
Facultad de Ingeniería

Procesador para control multiprotocolar

DOCUMENTO PRELIMINAR

Entregado como requisito para la obtención del título de
Ingeniero en Telecomunicaciones

Alan Cohn - Nº 137033
Rossana Morales - Nº 67756

Tutor:
Ing. Matias Nogueira

Setiembre 2006

Índice general

Índice de figuras

Índice de cuadros

Abstract

El proyecto consiste en la investigación, diseño y construcción de un sistema capaz de ofrecer el uso de determinados protocolos a través de una interfaz Ethernet. Se busca crear un dispositivo dedicado a intercomunicar conjuntos de protocolos incompatibles. Se implementará el control sobre cuatro protocolos distintos, siendo investigados específicamente los siguientes: IrDA, X10, RS232, RS485, RFI. El estudio concreto de los protocolos anteriormente mencionados corresponde a su gran inserción en el mercado de la ingeniería. El sistema constará de un bloque central y varios módulos de salida, cumpliéndose la condición que cada módulo de salida manejará y controlará un único protocolo.

Capítulo 1

Introducción

El sistema estará constituído por un bloque central y varios módulos de salida. Se buscará la escalabilidad y personalización del sistema apuntando al empleo de varios módulos de salida en forma conjunta, como también, a la utilización de diversos módulos que gobiernan el mismo tipo de protocolo. En base a las prestaciones del sistema el usuario determinará la cantidad de módulos a emplear. Los datos arribarán por la interfaz Ethernet y serán procesados de acuerdo a dicha personalización.

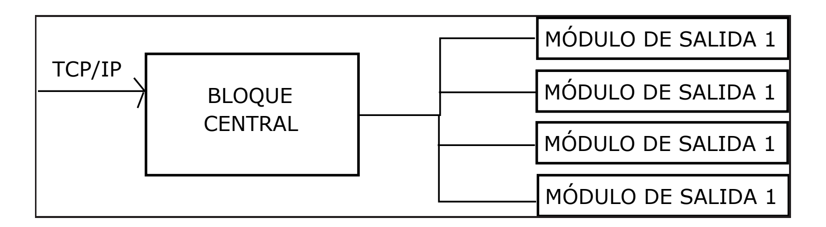


Figura 1.1: Diagrama de bloques

Se requiere un sistema robusto ante las inclemencias del ambiente de trabajo, capaz de operar en entornos húmedos, polvorientos y corrosivos, que esté protegido bajo aislamiento y por ende sin posibilidades de ventilación. Además, se contemplará su posible aplicación en entornos industriales donde se debe considerar su ubicación en lugares distantes, de dificil acceso y sometido a vibraciones. Asimismo, se desea que el sistema tenga una gran flexibilidad de aplicación, razón por la cual debe simplificarse su arquitectura. También, se considerará el estudio de la escalabilidad de dos sistemas iguales funcionando bajo las mismas directivas.

1.1. Motivación

Se ha optado por el desarrollo de un procesador de usos múltiples, el cual ofrecerá la integración de varias aplicaciones e interfaces en un mismo sistema.

Los productos disponibles en el mercado brindan soluciones para control distribuido, manejan distintos protocolos integrados dentro de un mismo sistema, pero no permiten la minimización y personalización del hardware a utilizar, entendiéndose por ésto, el uso y manejo de distintos módulos de salida de acuerdo a las necesidades de cada usuario. Ergo, la pretención de brindar al mercado un sistema de bajo costo, escalable y personalizables, que presente la posibilidad de elegir los protocolos a usar, con la finalidad de que el usuario pueda configurar su producto para cada situación particular.

1.2. Objetivos

- Establecer un protocolo general para comandar los distintos módulos de salida
- Proporcionar escalabilidad y versatilidad de módulos de salida
- Posibilidad de configurar y programar el sistema via Ethernet.
- Estudio de una estructura multiprotocolo.
- Pequeño, compacto y de fácil manejo.
- Obtener un prototipo funcionando.

1.3. Herraminetas de gestión de proyectos

Con el avance de la tecnología y la implementación de nuevas herramientas informáticas de software libre, se hace casi necesario la utilización de las mismas, ya que brindan facilidades que con ”lápiz y papel”son imposibles de sobrellevar, y además permiten una versatilidad superior a la hora de gestionar un proyecto en sí: documentación, tareas pendientes, seguimiento del proyecto, versionado de archivos guardados, etc.

1.3.1. TWiki

Con la finalidad de gestionar el seguimiento del proyecto se eligió una herramienta que debía cumplir con las siguientes condiciones:

- **guardar un registro de cambios:** almacenamiento de distintas versiones con los cambios hechos.
- **trabajar simultaneamente:** permitir la edición del contenido a varias personas en forma conjunta.

- **envío de avisos instantáneos**: notificación via E-mail cuando se modifica algún contenido.
- **acceso en forma remota:** autorizar la edición desde cualquier cualquier plataforma - Windows, Linux o Mac - o desde Internet.
- **documentar cada vez que se necesite:** servir como herramienta para escribir la documentación a medida que se avanza con el proyecto.
- **gestionar el seguimiento del proyecto:** tener una lista con tareas pendientes y resposabilidades, como también un registro con tareas terminadas.
- **tener reglas de escritura fáciles y simples:** documentar en el mismo lenguaje que se habla y no implementar reglas con nuevos formatos.
- **almacenar archivos:** guardar archivos de distinto tipo administrándolos ordenadamente.
- **jerarquizar y ordenar la información:** estructurar la información de forma legible y entendible.
- **poder insertar imágenes en los documentos:** documentar utilizando con imágenes

Tomando como base estas afirmaciones se decidió usar TWiki, una herramienta de la familia Wiki. Ésta aplicación permite tener un sitio web cuyas páginas son editables y accesibles por los usuarios que poseen el permiso para hacerlo. En este caso, el sitio es una web pública - accesible para culaquier persona -, pero la edición está limitada a los usuarios con clave de acceso.

Con esta aplicación es posible editar los distintos topics de forma conjunta sin que aparezcan conflictos, ya que los cambios quedan registrados y se posibilita la consulta a versiones anteriores brindando así, una gran flexibilidad de trabajo.

El TWiki está publicado en http://z0.saladeteletipos.com/twiki/bin/view /ProcesadorMultiprotocolar/WebHome

TWiki es una herramienta de software libre, escrita en Perl y distribuida con licencia GPL. Página oficial de TWIki - http://www.twiki.org

Ejemplo de utilización:

- http://z0.saladeteletipos.com/twiki/bin/view/ProcesadorMultiprotocolar /DocumentacionSetiembre - espacio donde se escribió ésta documentación.
- http://z0.saladeteletipos.com/twiki/bin/view/ProcesadorMultiprotocolar /ReunionesSemanalesTutor - registro de las reuniones con el tutor.

- http://z0.saladeteletipos.com/twiki/bin/view/ProcesadorMultiprotocolar /TareasPendientes - división de tareas generales y por persona.
- http://z0.saladeteletipos.com/twiki/bin/view/ProcesadorMultiprotocolar /DecisionSistemasEmbebidos - bitácora con argumentos de la razón de utilizar Rabbit.

1.3.2. Subversion — TortoiseSVN

Con el objetivo de disponer de un sistema de control de versiones se optó por una herramienta que debía cumplir las siguientes condiciones:

- **versionado de archivos:** un mismo archivo, con varias versiones, bajo el mismo nombre.
- **historial con cambios:** registro de los cambios realizados en cada archivo.
- **versatilidad en la aplicación:** posibilidad de modificar, mover, borrar cada uno de los elementos fácilmente.
- **mecanismos para el almacenamiento:** contener funciones que permitan especificar y detallar cambios.
- **herramientas de comparación:** desplegar sencillamente cambios en las versiones de archivos.

Tomando como base éstos conceptos se decidió emplear Subversion como sistema de control de versiones y TortoiseSVN como interface de acceso a Subversion.

Subversion se basa en la implementación de un Repositorio: un sistema centralizado donde se guarda información jerárquicamente en forma de "árbol", implementando directorios y archivos.

Cada usuario puede crearse una copia del Repositorio en su computadora y mediante las operaciones adecuadas tener siempre un historial con las distintas versiones de los archivos modificados.

Existen 2 operaciones básicas: svn commit y update. Svn commit se utiliza para confirmar los cambios y subirlos al repositorio, generando una nueva revisión de los archivos. Update se emplea para mantener una versión actualizada del Repositorio.

Esta herramienta permite mantener un órden de los archivos empleados, tanto para el desarrollo como para el producto final, de forma tal de no disponer de archivos innecesarios en los cuales cada versión está especificada con un cambio de nombre con la mera finalidad de identificarlo.

TortoiseSVN es la interfaz para Windows que permite realizar las siguientes aplicaciones:

- Subir archivos al Repositorio.
- Permitir notaciones adjuntas a la revisión cada vez que se realiza un commit.
- Actualizar el Repositorio local de cada computadora.
- Buscar diferencias entre versiones y señalarlas.
- Graficar utilización del Repositorio central por usuario y fecha

El Respositorio se encuentra publicado en http://z0.saladeteletipos.com/svn/proyecto pero no es de acceso público por razones de copyright; es necesario la autenticación correspondiente.

Subversion es una herramienta de software libre distribuida con licencia Apache/BSD. Web oficial de Subversión: http://subversion.tigris.org TortoiseSVN es una herramienta de software libre distribuida con licencia GPL. Web oficial de TortoiseSVN: http://tortoisesvn.tigris.org

La siguiente es un imagen con las funcionalidades de Subversion mediante TortoiseSVN:

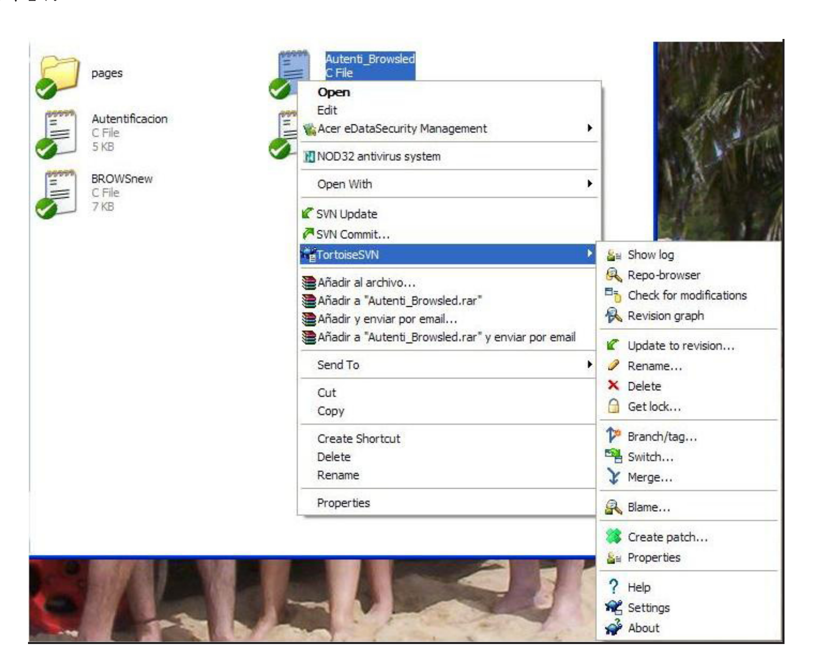


Figura 1.2: TortoiseSVN

Parte I

Sistemas Embebidos

Capítulo 2

Sistemas Embebidos con interfaz Ethernet

2.1. Introducción

Los microcontroladores de 8 bit generalmente son fáciles de usar pero habitualmente tienen poca memoria interna y su velocidad de procesamiento es limitada para algunas aplicaciones. Éstos microcontroladores pueden acceder a más memoria, pero a veces tampoco es suficiente. La conexión con chips de decodificación ocupa espacio de plaqueta y lleva tiempo de desarrollo y depurado, lo que implica un incremento de costos al producto final. Los microcontroladores de 16 bit son generalmente bastante más dificiles de usar, tienen mucha más memoria, pero a veces tampoco es suficiente; además suelen ser más caros y la conexión de memorias y perifericos a los buses suele traer problemas en ambientes donde el ruido y la interferencia están presentres. También necesitan de chips de decodificación con las caracteristicas asociadas que esto implica. Procesadores de 32 bit o más suelen ser más complejos de usar. No obstante, en velocidad y capacidad de memoria superan a las otras familias de microporcesadores. Los DSP (Digital Signal Processor, sistema basado en un microprocesador que posee un juego de instrucciones, un hardware y un software optimizados para aplicaciones que requieran operaciones numéricas a muy alta velocidad) son complejos de utilizar, el diseño del hardware suele ser muy complicado, necesitando chips muy rápidos que consumen más corriente, y algunas veces engorrosos circuitos de wait-states y memorias muy ágiles. Tendiendo un claro panorama de las metas propuestas, se comenzó por realizar una investigación con el objetivo de determinar .el corazón del sistema": encontrar el dispositivo adecuado de trabajo que funcionará como sistema central. No forma parte del proyecto resolver la comunicación entre la interfaz Ethernet y los distintos protocolos a nivel de sistema operativo, la solución a bajo nivel debe estar ya implementada. Por lo cual, se consideró únicamente el uso de sistemas embebidos con interfaz Ethernet ya que éstos cuentan con un de-

sarrollo avanzado y permiten lograr funcionalidades y aplicaciones sobre una base de directivas estables. Como criterios de evaluación se consideraron las prestaciones y características de las distintas placas, precios de venta y consideraciones elocuentes a la posibilidad de implementación y desarrollo sobre las mismas. El límite claro que se mantuvo fue no exederse de los recursos económicos brindados por la Universidad ORT para la realización del proyecto. Considerando las distintas opciones y la investigación llevada a cabo, se decidó por la compra del sistema embebido RCM3700 de la empresa Rabbit Semiconductors.

2.2. Placas Rabbit

Considerando las prestaciones brindidas y el equilibrio precio / funcionalidad se investigó el módulo RCM3700 y el RCM 3720 para su utilización.

2.2.1. RCM3700

- Módulo Ethernet de bajo costo basado en el procesador Rabbit 3000 a 22.1 MHz.
- Ethernet 10Base-T, RJ-45.
- Hasta 512K Flash / 512K SRAM.
- 1MB de Memoria Flash Serie.
- Conexión para batería externa.
- 33 E/S digitales / Bus de E/S alternado.
- 4 puertos serie (IrDA, HDLC, asynch, SPI).
- Tamaño: 7.5 cm x 3.0 cm x 2.2 cm

2.2.2. RCM3720

- Módulo Ethernet de bajo costo basado en el procesador Rabbit 3000 a 22.1 MHz.
- Ethernet 10Base-T, RJ-45.
- 512K Flash y 256K SRAM.
- 1MB de Memoria Flash Serie.
- Conexion para bateria externa.

- 33 E/S digitales / Bus de E/S alternado.
- 4 puertos serie (IrDA, HDLC, asynch, SPI).
- Tamaño: 7.5 cm x 3.0 cm x 2.2 cm

2.2.3. Dynamic C

Dynamic C es un software de desarrollo en lenguaje C industrialmente probado. Opera en cualquier PC bajo Windows 95, 98, NT, ME, 2000 y XP e incluye:

- Rápido compilador C que permite compilaciones de un solo paso, enlace y bajada al destino; debugger de completa prestación a nivel fuente y/o ensamblaje; varias funciones en las librerias de codigo fuentes y un editor fácil de usar.
- Debugging en tiempo real
- Un cable inteliligente, provisto con el kit de desarrollo, que conecta el puerto serial de la PC con uno de los puertos seriales Rabbit mientras el procesador esta corriendo en el sistema destino, quitando la necesidad de emuladores, simplificando así los desarrollos.
- El desarrollo de software con Dyanamic C es sencillo: código C, código Assembler o hasta código intermezclado de C con Assembler, pueden ser escritos, compilados y testeados sin la necesidad de salir del ambiente de desarrollo Dynamic C.

Prestaciones

Dynamic C está especificamente confeccionado para sistemas dedicados. Se puede compilar un programa y rutinas de librerias aplicables y bajar simultaneamente el codigo compilado hacia el sistema destino de manera muy rápida. Breakpoints, single stepping, observaciones de variables en un programa en ejecución, complejas expresiones visuales e impresión a la consola Dynamic C son todos soportados para ayudar al debbuging. También incluye poderosas extenciones de lenguaje para multitareas cooperativas, una gran velocidad en punto flotante y librerias matemáticas como también controladores para los perifericos integrados de Rabbit. Posee un stack TCP/IP completo. Sus capacidades incluyen Ethernet, PPP, cliente DHCP, http (incluye facilidades para SSI, rutinas CGI, cookies y autenticaciones básicas), SMTP, POP3, FTP y TFTP.

Producto	Descripción	Precio en U$S por unidad
RCM3700	512 KB de RAM, 512 KB de Flash	59
RCM3720	512 KB de RAM, 512 KB de Flash	49
Development Kit	con placa RCM3700	299
Develoment Kit	con placa RCM3720	299

Cuadro 2.1: Precios Placas Rabbit

Características

- Veloz compilador C con compilación de un solo paso, enlace y descarga a destino
- Completas prestaciones a nivel fuente y/o Assembler para debugger
- Cientos de funciones en librerías de código fuente.
- Entorno de facil manejo

2.2.4. Precios

Los precios fueron obtenidos directamente de la fábrica de Rabbit Semiconductors http://www.rabbitsemiconductor.com.

2.3. Placas Digi

2.3.1. ConnectCore 7U

Considerando las prestaciones brindidas y el equilibrio precio / funcionalidad se investigó únicamente el módulo ConnectCore 7U para su utilización.

2.3.2. Prestaciones

- Procesador de 32 bits
- Reloj de 55 MHz.
- 16 MB de memoria RAM
- 8 KB de memoria EEPROM
- Hasta 8 MB de memoria flash
- Placa Ethernet 10/100 base-T
- 2 puertos USART y SPI

- Trabaja sobre un sistema operativo Linux
- 2 timers de 27 bits.
- Interface JTAG on-board
- Tamaño: 6.28 cm x 1.85 cm x 1.04

2.3.3. Consideraciones

Para desarrollar sobre la placa se necesita la plataforma LxNETES 2.3 (linux) o la plataforma NET+OS 6.0. Cada una de estas plataformas está desarrollada en un development kit.

- El LxNETES 2.3 incluye:
  - Programación en flash
  - UClinux kernel v2.4.22
  - GNU development tool
  - Sistema de archivos que soporta CRAMFS, JFFS2, NFS.
  - Servicio PPP
  - Cliente Samba
  - Web server embebido
  - Debugging via Ethernet o serial
  - Códigos de ejemplo y documentación
- El NET+OS 6.0 incluye:
  - Debugging via JTAG
  - GNU development tool
  - Stack de protocolos TCP/IP
  - Asignación IP universal (Static IP, DHCP, BOOTP, auto -ip)
  - Web server embebido
  - SSL/TLS w/DES, 3DES, AES
  - Compilador HTML a C
  - Códigos de ejemplo y documentación

Los kits de desarrollo no incluyen la placa, por lo cual para el desarrollo de una aplicación específica se debe comprar la placa más el kit de desarrollo específico.

Item	Descrip.	Precio U$S por unidad
ConectorCore 7U	16MB SDRAM, 2MB de Flash	99
	16 MB de SDRAM, 8 MB de Flash	159
	32 MB de SDRAM, 16 MB de Flash, 180 Mhz	175
Development Kits	LxNETES Linux	299
	Net+OS	1495

Cuadro 2.2: Precios Placas Digi

2.3.4. Precios

Los precios fueron obtenidos del catálogo de http://www.mouser.com/digi - Mouser Electronic - (Empresa distribuidora de productos Digi en Estados Unidos).

2.4. Placas PC104

2.4.1. TS-7200 ARM

Considerando las prestaciones brindidas y el equilibrio precio / funcionalidad se investigó únicamente la placa TS-7200 ARM SBC para su utilización.

2.4.2. Prestaciones

La TS-7200 es compacta, con características completas de Single Board Computer (SBC) basada sobre la CPU del Cirrus EP9302 ARM9. Las características del EP9302 y un avanzado procesador de 200 Mhz con una unidad de gerencia de memoria (MMU) que permite soportar sistemas operativos de alto nivel, tales como Linux, Windows CE y otros sistemas operativos embebidos. El propósito general del procesador es proveer un conjunto estándar de periféricos en la placa y un conjunto completo de sistemas Technologic que ofrecen los periféricos vía el estándar Bus PC/104.

2.4.3. Características:

- Sistema operativo TS-Linux Embedded instalado
- 200 MHz ARM9 CPU con MMU
- 8 MB on-board Strata Flash (Bootear a Linux)
- 32 MB memoria RAM
- True IDE Compact Flash socket (para memoria Flash adicional)

- 2 USB 2.0 compliant Full Speed host (OHCI) ports - 12 Mbps máximo
- 2 puertos seriales (hasta 230 Kbaud)
- 10/100 Megabit Ethernet port
- 20 líneas E/S digitales
- Watchdog Timer
- Bus de expansión PC/104
- SPI bus interface
- Intefaz para LCD
- Single +5VDC power supply @ 450 mA
- Tamaño: 9.5 cm x 11.25 cm.

2.4.4. Consideraciones

Para desarrollar sobre la placa se necesita un Development Kit específico:

- Caracteristicas:
  - 256 MB or 512 MB Flash drive
  - Rutinas de testeo de hardware, código fuente y ejemplos.
  - USB Compact Flash reader.
  - 5 VDC power supply
  - Cable adaptador de 10 pines a DB-9.
  - Cables para conexiones DIO, LCD, Keypad, etc.
  - Development CD con fuentes, manuales y ejemplos de código

El kit de desarrollo no incluye la placa, por lo cual para el desarrollo de una aplicación específica se debe comprar la placa más el kit de desarrollo.

Precios
Los precios fueron obtenidos del catálogo de http://www.embeddedarm.com/epc/ts7200-spec-p.php#7200pricing - Technologic Systems - (Empresa distribuidora de productos PC104 en Estados Unidos).

Precios	Descripción	Precio en U$S por unidad
TS-7200 SBC	32 MB de RAM, 8 MB de Flash	149
TS 7200 SBC	32 MB de RAM, 16 MB de Flash	165
	usb 802.11g wireless network interface	35
	512 MB Compact Flash Card	105

Cuadro 2.3: Precios Placas PC104

2.5. Comparación entre placas

Valorando las distintas opciones y habiendo visto las diferentes características de las placas se tomaron ciertos criterios para establecer la comparación entre las mismas: precio, dimensiones, características y soporte ante fallas.

Precio
Las placas Rabbit son las más baratas, no solamente por el costo de la placa en sí, sino también el kit de desarrollo de Rabbit incluye la placa. Vale recalcar que ésta es una gran consideración ya que no es posible desarrollar alguna solución sin un development kit; además las limitantes económicas del proyecto están definidas por la Universidad ORT.

Dimensiones
Como la finalidad del sistema embebido consiste en ser el çorazón del sistema"vale establecer que sus dimensiones deberán ser pequeñas. Tanto las placas Rabbit como Digi poseen dimensiones pequeñas; PC104 posee dimensiones mayores.

Características
Cuanto mayores son las capacidades de las placas, mayor es el provecho a sacarles. Visto y considerando los objetivos, no es necesario el desarrollo sobre una SBC, las cuales poseen prestaciones excesivas para los alcances de este proyecto. Cabe resaltar que tanto las placas Digi como las PC104 corren sobre un sistema operativo basado en Linux. Rabbit es un microcontrolador que no está desarrollado sobre un sistema operativo booteable. Las capacidades de memoria de Digi como las de PC104 son ampliamente superables a Rabbit, pero poseer una capacidad de memoria excesiva para el desarrollo del proyecto no es una apreciación valorable en la evaluación de las placas. La velocidad de procesamiento es una consideración importante, el procesador de Rabbit trabaja a una velocidad de 22.1 MHz, Digi a 46 o 55 MHz y las PC104 a 200MHZ. Cuanto mayor velocidad más rápido se ejecutan las instrucciones, pero 22.1 MHZ es una velocidad de procesamiento bastante considerable. La diferencia entre las dos placas Rabbit consiste en la capacidad de la memoria

RAM del sistema, en el resto poseen las mismas prestaciones.

Soporte
Buscando en Internet es posible encontrar aplicaciones desarrolladas para las tres placas. Tanto Rabbit como Digi ofrecen como valor agregado a su producto un support que brinda ayuda las 24 hrs. del día y puede ser contactado tanto por e-mail como por teléfono. A la hora de considerar el desarrollo de un producto es bueno valorar la posibilidad externa de ayuda ya que cualquier impedimento en el desarrollo puede acarrear una pérdida de tiempo considerable en el avance del proyecto. El sitio web http://www.pc104.org ofrece ayuda a desarrolladores paras las placas PC104. Comparando, Digi y Rabbit ofrecen un servicio más serio que PC104, ya que dedican recursos de sus compañías para brindar un Tech Support amigable y accesible las 24hrs.

2.5.1. ¿Por qué Rabbit?

Los procesadores de la linea Rabbit unifican muchas de ventajas:

- El Rabbit es un microprocesador: su bus es accesible.
- Su bus de direcciones es de 20bits, lo que implica 1MB de direccionamiento posible.
- Su bus de datos es de 8 bits: menos líneas de conexión.
- Su arquitectura es compatible con el Z-80: conocido, con gran cantidad de software disponible.
- Es un microcontrolador: I/O ports, USARTs (seriales), timers, WDT (watch-dog), RTC (real-time clock), todos incluídos en un chip.
- Chip Selects que eliminan la glue-logic para memoria, generación de 0 a 4 wait-states.
- I/O strobes que eliminan la glue-logic para I/O, con generación de 0 a 15 wait-states.
- Serial boot: siempre es posible cargar la última versión, sin parches.
- Slave port: facilita la interconexión de procesadores, incluso puede bootear del slave port.
- Su performance es comparable a la de muchos DSP del mercado.
- El kit de desarrollo viene acompañado por un potente entorno de programación C con bibliotecas de funciones, soporte multitarea, TCP/IP (sin regalías), e ICD (In-Circuit Debugging).

- Las dimensiones de la placa son pequeñas.

Core Modules

Si bien el procesador no tiene memoria interna, en vez de comprar el procesador por separado y diseñar la placa de circuito impreso existe la posibilidad de adquirir los core-modules: módulos pre armados con cierta capacidad de memoria ya instalada, pines para la conexion con el mundo exterior, controlador Ethernet y jack RJ-45. Éstos módulos son un recurso probado en ambientes conflictivos, económicos y de fácil implementación: pueden soldarse o conectarse con un zócalo a la placa principal, en el cual el diseñador no necesita preocuparse por los buses ni por el timming, sino que puede trabajar con las entradas y salidas como en cualquier microcontrolador. En caso de necesitar mas memoria para el proyecto, en vez de portar el diseño a un controlador de gama superior, simplemente se puede remplazar el modulo por uno de mayor capacidad.

TCP/IP sobre Ethernet

Ésta es la caracteristica más interesante del producto. Cualquier desarrollador podría: elegir un procesador, desarrollar la interfaz Ethernet, conseguir un stack TCP/IP reducido que ha sido recortado y compactado para caber en memoria; con la placa Rabbit sólo es necesario conectar el patch-cord y comenzar a desarrollar.

Dynamic C

El compilador es una implementación de C con el agregado de funciones específicas para Rabbit, soporte multitarea cooperativo, y bibliotecas de funciones que resuelven gran cantidad de tareas en aplicaciones típicas. Permite debugging en circuito a nivel C o a nivel assembler, seleccionable por el usuario. Puede ejecutarse instrucción por instrucción, con feedback en pantalla, o ponerse breakpoints. También incluye soporte para TCP/IP sin ningún problema.

Funciones provistas por Dynamic C:

- Funciones standard de C, incluyendo coma flotante (floating point) y funciones trascendentes.
- Multitarea cooperativo
- Interfaz I2C
- Interfaz con GPS receivers en NMEA-0183
- FFT (Fast Fourier Transforms)

- Interfaz SPI
- RTC (Real Time Clock)
- Interfaz serie (stream oriented, con flow control, circular buffers, frame oriented)

Funciones adicionales para TCP/IP, incluídas en Dynamic C:

- DHCP client
- HTTP server con soporte SSI y CGI
- FTP server/client
- TFTP server/client
- ICMP (ping)
- POP3 client
- SMTP client
- Acceso socket level a UDP y TCP

En resumen, se trata de un micro de propositos generales, con gran cantidad de puertos de I/O, posibilidad de conexion al bus, alta velocidad de operación, amplia capacidad de RAM y Flash, programación en C, con modulos pre armados que facilitan la velocidad de desarrollo y un gran stack de directivas TCP/IP para implementar.

Capítulo 3

Protocolos

3.1. Introducción

El intercambio de información y datos, como también la comunicación entre sistemas es la base de toda red. En ingeniería, al lenguaje que emplean los distintos elementos que componen una red se lo denomina ”protocolo”. Como definición más formal, se considera un protocolo de comunicación al conjunto de reglas que controlan la secuencia de mensajes que ocurren durante una comunicación entre entidades que forman una red (según http://es.wikipedia.org/wiki/Protocolo_de_red). Las entidades son vistas como dispositivos electrónicos, automatísmos o bien software que interactúan en la red. Los protocolos implementados en sistemas de comunicación que tienen gran impacto, suelen convertirse en estándares. Existen consorcios empresariales, que tienen como propósito proponer recomendaciones de estándares que se deben respetar para asegurar la interoperabilidad de los productos. En este capítulo se estudian los distintos protocolos que forman parte del proyecto, los cuales corresponden a especificaciones de capa física y aplicación. Dichos protocolos, no solamente se emplearon en la etapa de implementación sino también en el análisis y la investigación correspondiente.

3.2. Protocolo OSGI

La Open Service Gateway Initiative (OSGI) es una asociación de empresas creada en marzo de 1999 con el objetivo de definir un estándar abierto para el desarrollo y diseño de pasarelas residenciales **1** que sean capaces de brindar múltiples servicios en el mercado residencial y automotriz. OSGI ofrece una arquitectura completa y una solución extremo a extremo que cubre todas las necesidades del proveedor de servicios, del cliente y de los distintos dispositivos instalados en la vivienda.

3.2.1. Descripción

No escoge una única tecnología de conexión en red para los múltiples dispositivos de la vivienda, siendo su propósito definir una interface común para todos ellos dejando la responsabilidad a los fabricantes de construir controladores adecuados. Define una plataforma de software, basada en Java, independiente del hardware. Los componentes de software son librerías o aplicaciones que dinámicamente pueden descrubrir y usar otros protocolos. Las especificaciones de OSGI son extensamente aplicables ya que conforman una capa pequeña que permite múltiples componentes basados en Java™ para cooperar eficientemente en una sola máquina virtual de Java (JVM).

Las pasarelas OSGI pueden utilizar distintas tecnologías:

- conexiones inalámbricas: IrDa, HomeRF, IEEE 802.11x., Bluetooth, etc.
- cables telefónicos: HomePNA, etc.
- redes de baja tensión: HomePlug, Lonworks, EIB/KNX, etc.
- otras conexiones: Ethernet, USB, etc.
- distintos protocolos: HAVI, UPnP, Jini, etc.

3.2.2. Características

Detalla un conjunto de APIs (Application Program Interfaces) que son el principal soporte de los servicios. Los rasgos principales de la especificación son:

- **Estandarizada:** poseer una plataforma común para los fabricantes de equipos y los proveedores de servicios e impedir que un único fabricante monopolice el mercado.
- **Abierta:** no se define ninguna estructura de red domótica **2** ni se decreta el uso de ningún protocolo ni tecnología en concreto. La única condición es que las tecnologías deben ser compatibles con las APIs predefinidas.
- **Fiable:** debe funcionar las 24 hrs. del día sin caídas del sistema.
- **Segura:** obliga un nivel de seguridad e integridad con el objetivo que los proveedores ofrezcan múltiples servicios sobre la misma plataforma sin interferirse entre ellos.
- **Escalable:** la operación de las distintas pasarelas debe ser flexible, personalizable y escalable acorde a las nuevas necesidades del proveedor del sistema.

3.2.3. Arquitectura

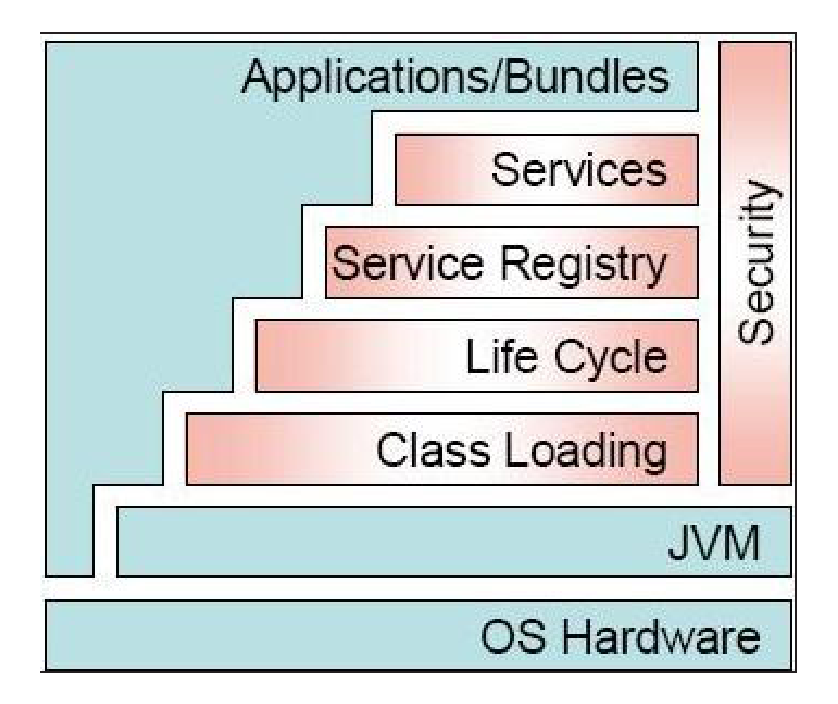


Figura 3.1: Diagrama de bloques

Para comprender la implementación del framework es necesario conocer algunos conceptos:

- **Bundle** - es un archivo JAR de Java que corresponde al mecanismo utilizado para distribuir e instalar aplicaciones y servicios en la plataforma.
- **Service** - dentro del la plataforma OSGi, se considera que un servicio es un objeto registrado como proveedor de una determinada interfaz.
- **Activator** - es la clase encargada de gestionar el ciclo de vida de un Bundle. Posee métodos start y stop que son invocados cuando el Bundle comienza y se detiene, respectivamente.
- **Wire** - es un canal de comunicación entre dos servicios siguiendo una estrategia productor/consumidor.

3.2.4. Funcionamiento

Los bundles pueden instalarse remotamente, comenzando, parando, puesto al día y desinstalación sin requerir un reboot y proporcionan una determinada funcionalidad a otros paquetes o directamente al usuario final. Los bundles residen sobre un elemento central llamado Plataforma de Servicios OSGI situada

en la red local y conectada al proveedor de servicios a través de una pasarela en la red del operador; este elemento permite la interacción entre redes de dispositivos que empleen distintas tecnologías para la comunicación. El registro de servicio actúa como un servicio de directorios en el que los bundles se registran y pueden localizar a otros bundles para articular otros servicios. En la especificación se definieron APis básicas: logging, servidor HTTP y DAS (Device Access Specification). Con el DAS se definen los bundles de red - encargados de descubrir nuevos dispositivos y protocolos empleando el protocolo de descubrimiento correspondiente -, una vez alcanzada esta información, deben adquirirse del proveedor de servicios el bundle de dispositivo correspondiente al dispositivo declarado que se instalará en la plataforma y se registrará en el Registro de Servicios OSGI, creando la asociación correspondiente con el proveedor del servicio para que puedan interactuar.

- *1* - dispositivos que conectan las infraestructuras de telecomunicaciones (datos, control, automatización, etc.) de la vivienda a una red pública de datos. Según http://www.casadomo.com/noticiasDetalle.aspx ?c=49&m=15&idm=60&pat=14&n2=14

- *2* - conjunto de sistemas automatizados de una vivienda que aportan servicios de gestión energética, seguridad, bienestar y comunicación, y que pueden estar integrados por medio de redes interiores y exteriores de comunicación, alambradas o inalámbricas. Según http://es.wikipedia.org

3.3. Protocolo HTTP

El Protocolo de Transferencia de HiperTexto (Hypertext Transfer Protocol) es un sencillo protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP. Desde el punto de vista de las comunicaciones, está soportado sobre los servicios de conexión TCP/IP, y funciona de la siguiente misma forma: un proceso servidor escucha en un puerto de comunicaciones TCP (por defecto, el 80), y espera las solicitudes de conexión de los clientes Web. Una vez que se establece la conexión, el protocolo TCP se encarga de mantener la comunicación y garantizar un intercambio de datos libre de errores.

HTTP se basa en sencillas operaciones de solicitud/respuesta. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud. El servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado. Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto Web (documento HTML, fichero multimedia o aplicación CGI) es conocido por su URL.

3.3.1. Etapas de una transacción HTTP

Para profundizar más en el funcionamiento de HTTP, veremos un caso particular de una transacción HTTP. Cada vez que un cliente realiza una petición a un servidor, se ejecutan los siguientes pasos:

- Un usuario accede a una URL, seleccionando un enlace de un documento HTML o introduciéndola directamente en el campo Location del cliente Web.
- El cliente Web descodifica la URL, separando sus diferentes partes. Así identifica el protocolo de acceso, la dirección DNS o IP del servidor, el posible puerto opcional (el valor por defecto es 80) y el objeto requerido del servidor.
- Se abre una conexión TCP/IP con el servidor, llamando al puerto TCP correspondiente.
- Se realiza la petición. Para ello, se envía el comando necesario (GET, POST, HEAD,...), la dirección del objeto requerido (el contenido de la URL que sigue a la dirección del servidor), la versión del protocolo HTTP empleada (casi siempre HTTP/1.0) y un conjunto variable de información, que incluye datos sobre las capacidades del browser, datos opcionales para el servidor,....
- El servidor devuelve la respuesta al cliente. Consiste en un código de estado y el tipo de dato MIME de la información de retorno, seguido de la propia información.
- Se cierra la conexión TCP.

3.3.2. Ejemplo de un diálogo HTTP

Para obtener un recurso con el URL http://www.tuhost.example/index.html
Se abre un socket con el host www.tuHost.example, puerto 80 que es el puerto por defecto para HTTP. Se envía un mensaje en el estilo siguiente:

GET /index.html HTTP/1.0
From: yo@miHost.example
User-Agent: HTTPTool/1.0
Línea en blanco

La respuesta del servidor está formada por encabezados seguidos del recurso solicitado, en el caso de una página web:

HTTP/1.0 200 OK
Date: Fri, 31 Dec 2003 23:59:59 GMT
Content-Type: text/html
Content-Length: 1221
<html>
<body>
<h1>Página principal de tuHost</h1>
(Contenido)
.
.
.
</body>
</html>

Al recibirse la respuesta, el servidor cierra la comunicación. Cabe señalar que los principales navegadores web no muestran al usuario los encabezados HTTP del recurso. Para visualizar tales encabezados pueden utilizarse herramientas conocidas genéricamente como ”visores HTTP”.

3.4. Protocolo SMTP

SMTP (Simple Mail Transfer Protocol o protocolo simple de transferencia de correo electrónico) es un protocolo de red basado en texto que se utiliza para el intercambio de mensajes de correo electrónico entre computadoras o distintos dispositivos. Se basa en el modelo cliente-servidor, donde un cliente envía un mensaje a uno o varios receptores. En el conjunto de protocolos TCP/IP, el SMTP va por encima del TCP, usando normalmente el puerto 25 en el servidor para establecer la conexión.

3.4.1. Ejemplo de una comunicación SMTP

En primer lugar se ha de establecer una conexión entre el emisor (cliente) y el receptor (servidor). Esto puede hacerse automáticamente con un programa cliente de correo o mediante un cliente telnet. El ejemplo a continuación muestra una conexión típica. con la letra C se nombra al cliente y con S al servidor

S: 220 Servidor ESMTP
C: HELO
S: 250 Hello, please meet you
C: MAIL FROM: yo@midominio.com
S: 250 Ok
C: RCPT TO: destinatario@sudominio.com

S: 250 Ok
C: DATA
S: 354 End data with <CR> <LF>.<CR> $< LF >$
C: Subject: Campo de asunto
C: From: yo@midominio.com
C: To: destinatario@sudominio.com
C:
C: Hola,
C: Esto es una prueba.
C: Adios.
C: .
S: 250 Ok: queued as 12345
C: quit
S: 221 Bye

En el protocolo SMTP todas las órdenes, réplicas o datos son líneas de texto, delimitadas por el carácter . Todas las réplicas tienen un código numérico al comienzo de la línea.

Cuando un cliente establece una conexión con el servidor SMTP, espera a que éste envíe un mensaje “220 Service ready” o “421 Service non available” Se envía un HELO desde el cliente. Con ello el servidor se identifica. Esto puede usarse para comprobar si se conectó con el servidor SMTP correcto. El cliente comienza la transacción del correo con la orden MAIL. Como argumento de esta orden se puede pasar la dirección de correo al que el servidor notificará cualquier fallo en el envío del correo. El servidor responde “250 OK”. Ya le hemos dicho al servidor que queremos mandar un correo, ahora hay que comunicarle a quien. La orden para esto es RCPT TO:¡destino@host¿. Se pueden mandar tantas órdenes RCPT como destinatarios del correo queramos. Por cada destinatario, el servidor contestará “250 OK” o bien “550 No such user here”, si no encuentra al destinatario. Una vez enviados todos los RCPT, el cliente envía una orden DATA para indicar que a continuación se envían los contenidos del mensaje. El servidor responde “354 Start mail input, end with .” Esto indica al cliente como ha de notificar el fin del mensaje. Ahora el cliente envía el cuerpo del mensaje, línea a línea. Una vez finalizado, se termina con un . (la última línea será un punto), a lo que el servidor contestará “250 OK”, o un mensaje de error apropiado. Tras el envío, el cliente, si no tiene que enviar más correos, con la orden QUIT corta la conexión. También puede usar la orden TURN, con lo que el cliente pasa a ser el servidor, y el servidor se convierte en cliente. Finalmente, si tiene más menajes que enviar, repite el proceso hasta completarlos.

En el ejemplo pueden verse las órdenes básicas de SMTP:

- HELO, para abrir una sesión con el servidor

- MAIL FROM, para indicar quien envía el mensaje
- RCPT TO, para indicar el destinatario del mensaje
- DATA, para indicar el comienzo del mensaje, éste finalizará cuando haya una línea únicamente con un punto.
- QUIT, para cerrar la sesión

Las respuestas que da el servidor pueden ser de varias clases:

- 2XX, para una respuesta afirmativa
- 3XX, para una respuesta temporal afirmativa
- 4XX, para una respuesta de error, pero se espera a que se repita la instrucción
- 5XX, para una respuesta de error

Una vez que el servidor recibe el mensaje finalizado con un punto puede bien almacenarlo si es para un destinatario que pertenece a su dominio, o bien retransmitirlo a otro servidor para que finalmente llegue a un servidor del dominio del receptor.

3.4.2. Formato del mensaje

Como se muestra en el ejemplo anterior, el mensaje es enviado por el cliente después de que éste mande la orden DATA al servidor. El mensaje está compuesto por dos partes: Cabecera: en el ejemplo las tres primeras líneas del mensaje son la cabecera. En ellas se usan unas palabras clave para definir los campos del mensaje. Éstos campos ayudan a los clientes de correo a organizarlos y mostrarlos. Los más típicos son subject (asunto), from (emisor) y to (receptor). Éstos dos últimos campos no hay que confundirlos con las órdenes MAIL FROM y RCPT TO, que pertenecen al protocolo, pero no al formato del mensaje. Cuerpo del mensaje: es el mensaje propiamente dicho. En el SMTP básico está compuesto únicamente por texto, y finalizado con una línea en la que el único carácter es un punto.

3.5. Protocolo RS232

Es un estándar desarrollado en los años 60 por la EIA (Asociación de Industrias Electrónicas) conjuntamente con los laboratorios Bell y los fabricantes de equipos con la finalidad inicial de intercomunicar un equipo terminal de datos (DTE) y un equipo de comunicación de datos (DCE), empleando un intercambio de datos binarios en forma serial.

Desde su introducción, el EIA introdujo tres modificaciones, la más reciente la EIA-RS232-F en 1997.

El estándar define características de la señal eléctrica, características mecánicas de la conexión, descripción funcional de los circuitos de intercambio y secuencia de procedimientos.

3.5.1. Características

Características de la señal eléctrica

La interfaz emplea conexiones elécticas no balanceadas - el nivel de señal es relativo a la tierra de señal -, son más susceptibles al ruido y emplean velocidades menores a las conexiones balanceadas - un par de hilos por señal, no referido a tierra -.

Es una interfaz de bajo voltaje que opera entre -15 V. y +15 V. donde:

- Voltajes entre - 3 V. y - 15 V. corresponden a un ”1”lógico.
- Voltajes entre + 3 V. y + 15 V. corresponden a un ”0”lógico.
- Voltajes entre - 3 V. y + 3 V. corresponden a una región de transición o seguridad.

La región de transición funciona como un margen donde no se define un estado lógico ya que los cables pueden estar sometidos a ruidos e interferencias eléctricas que pueden llevar a transiciones de estado no deseadas: si aumenta la velocidad de transmisión la señal se vuelve susceptible a pérdidas de voltaje por efecto de las altas frecuencias, motivadas por la resistencia, inductancia y capacidad del cable como línea de transmisión, y aumentan con la longitud del cable. El ancho de la región de seguridad determina el margen de ruidos y limita directamente la velocidad máxima de transmisión de datos sin pérdidas.

Características mecánicas de la conexión

Establece que el DCE dispondrá de un conector hembra y el DTE de un conector macho donde se especifica la asignación de números de identificación a cada pin del conector. El tipo y medidas de los conectores son establecidos por la ISO (International Standars Organization). Los conectores más utilizados son los de nueve pines (DB-9) y veinticinco pines (DB-25).

Características funcionales de la conexión

El RS232 es una interfaz serie, lo cual implica que entrega la información en forma secuencial, bit por bit.

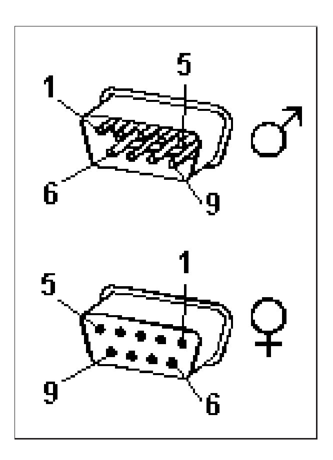


Figura 3.2: Conectores

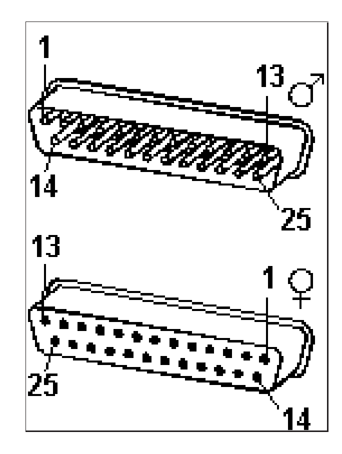


Figura 3.3: Conectores DB25

Existe un juego secundario de señales que incluyen las señales de control, que se utiliza para enviar información de configuración al extremo remoto, solicitar retransmisiones y funciones de control. Para asegurar la fiabilidad de éste medio típicamente trabajan a velocidades muy bajas.

- Pin 14 - Secondary Transmitted Data (STxD)
- Pin 16 - Secondary Received Data (SRxD)
- Pin 19 - Secondary Request to Send (SRTS)
- Pin 13 - Secondary Clear to Send (SCTS)

Las señales en un DTE y un DCE deben tener el mismo nombre, aunque el pin 2 del DTE "transmita" y el pin 2 del DCE "reciba", ambas señales son Transmit Data.

DB25	DB9	RJ45	Nombre	Función
2	3	6	TxD	Transmisión de datos (out)
3	2	5	RxD	Recepción de datos (in)
4	7	8	RTS	Pedido de envio (out)
5	8	7	CTS	Dispuesto a enviar (in)
6	6	1	DSR	Dispositivo de datos listo (in)
7	5	4	GND	(Común ground)
8	1	2	CD	Detección de portadora (in)
20	4	3	DTR	Terminal de datos lista (out)
22	9	1	RI	Indicador de llamada (in)
24	-	-	RTxC	Reloj de tarnasmisión/recepción (out)

Cuadro 3.1: Conexiones

Secuencia de procedimientos

Su funcionamiento puede ser sincrónico como asíncrono.

Transmisión síncrona
Se requieren señales de timming. Los pines 15, 17 y 24 se utilizan únicamente en modo sincrónico

Transmisión asíncrona
Es necesario el empleo de bits de stop y start. El comienzo de flujo de datos se reconoce porque la señal pasa de "marca.[a] .[es]pacio". Los bits de paridad se utilizan con el fin de verificar la integridad de la conexión.

Tipos de paridad:

- No Parity (sin paridad) - No se transmite bit de paridad
- Even Parity (paridad "par") - El bit de paridad es uno (1) si el caracter lleva un cantidad par de unos.
- Odd Parity (paridad "impar") - El bit de paridad es uno (1) si el caracter lleva una cantidad impar de unos.
- Mark Parity (paridad de "marca") - El bit de paridad siempre es uno
- Space Parity (paridad de .[es]pacio") - El bit de paridad siempre es cero

El estándar no establece como representar caracteres (7 u 8 bits es la forma más común, pero podrían ser 5 0 6). Cuando no se envían datos la señal se debe mantener en estado de marca (un "1"lógico). El control de flujo puede realizarse por hardware (RTS/CTS) o sofware (Xon/Xoff).

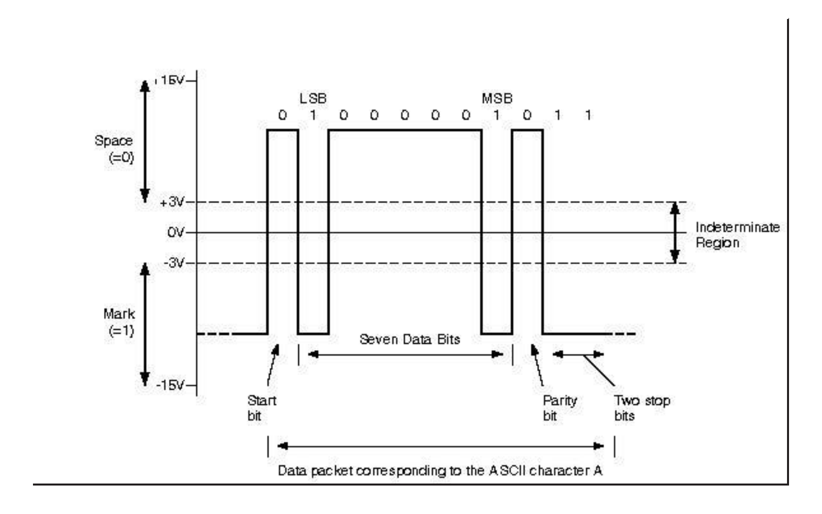


Figura 3.4: T Asincrona RS232

Por Software: El carácter Xoff (ASCII 19) es utilizado por el receptor para indicar que su buffer está lleno y el emisor debe esperar. Cuando vuelva a tener espacio en el buffer, enviara el carácter Xon (ASCII 17), informando que puede volver a trasmitir. Esto economiza cables en la interconexión, pero ocupa espacio en el canal.

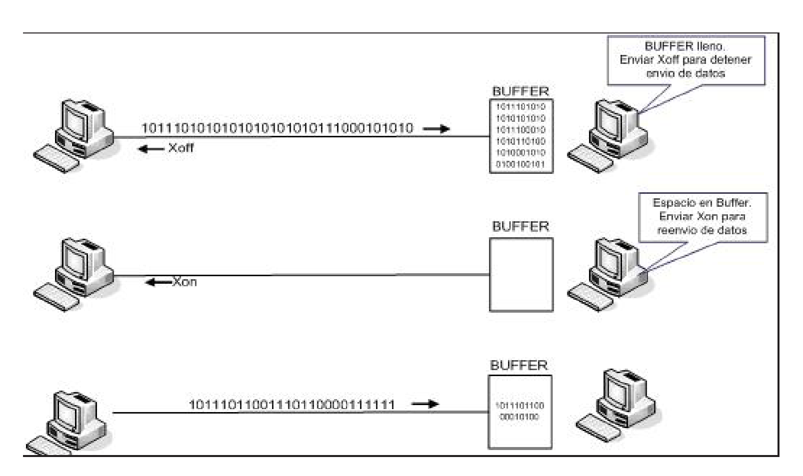


Figura 3.5: Control de flujo RS232

Por Hardware: Requiere que entre los dispositivos se conecten dos hilos: RTS y CTS. Cuando el buffer del receptor se llena le dice al transmisor que espere, desactivando la señal CTS. Cuando vuelva a tener espacio en el buffer, activa nuevamente el CTS para decir que está nuevamente listo.

Tiempos

Para las señales de control el tiempo de transición por la región de seguridad debe ser menor a 1 mseg. Para las señales de datos y reloj el tiempo de

transición por la región de seguridad deben ser:

- menores a 1 mseg para señales con períodos mayores a 25 mseg.
- 4 % del período para señales con período entre 125 microseg. y 25 mseg.
- menores a 5 microseg. para señales con período menor a 125 microseg.

3.5.2. Inconvenientes y problemas

- Funcionamiento incorrecto de comunicación (DTE-DCE) para el cable en uso dado por la inversión de las líneas de transmisión y recepción así como también líneas de handshaking.
- Tipo de conexión incorrecta por error de género o por configuración de pines.
- No se pueden lograr velocidades de transmisión mayores a 64 kbps por la modalidad de trabajo no balanceada.
- Puede comunicarse con un solo dispotivo por conexión.

Capítulo 4

Implementación

4.1. Introducción

Considerando haber determidado la magnitud del proyecto, realizado el análisis y la investigación correspondiente a la base teórica necesaria para corresponder a la etapa de implementación, en este capítulo se documentan las distintas decisiones tomadas y el avance del proyecto. En este capítulo se explican las decisiones tomadas sobre:

- protocolos a implementar para los distintos módulos de salida
- estrategia de programación de la placa Rabbit
- pruebas y avances

Cabe destacar que la etapa de análisis e investigación no finalizó, ya que habiendo tomado las distintas decisiones con el correr del desarrollo del proyecto van surgiendo determinados inconvenientes que requieren de un nuevo estudio de las condiciones existentes y soluciones determinadas.

4.2. Decisiones sobre los protocolos a implementar

Como primera medida se investigó el protocolo OSGI y se descubrió que correspondía plenamente con los objetivos trazados para este proyecto: establecer la comunicación correspondiente entre los distintos módulos de salida y la red externa, tomando como base una plataforma de software ya resuelta, lo cual denotaría un gran avance del proyecto. No se considera la implemntación del protocolo OSGI tal cual está desarrollado ya que la placa RCM3700 no corre sobre un sistema operativo, por lo cual es posible instalarle una Java Virtual Machine. La única forma de implementar OSGI sería implementar una

máquina virtual capaz de interpretar byte core de Java, lo cual no es muy eficiente para la placa Rabbit ni tampoco forma parte de los objetivos la árdua programación en Java. Previamente, habiendo realizando un balance en base a prestaciones/costos y tomando exhaustivamente las limitaciones económicas del proyecto se consideró el no comprar una placa de desarrollo con un sistema operativo basado en Linux o Windows. Valorando su uso y las distintas aplicaciones que comparten una interfaz serial se comenzó por la implementación del protocolo RS232. Considerando el claro objetivo del proyecto de comandar los distintos módulos de salida por Ethernet se resolvió utilizar el protocolo HTTP para correr un servidor web dentro de la placa. Las páginas web con las cuales se realizaron las distintas pruebas se programaron en lenguaje HTML. El protocolo SMTP fue motivo de estudio como una prestación del módulo rabbit y su facilidad para enviar correos electrónicos utilizando distintas librerías. Se estudió con la finalidad de implementar el envío de correos electrónicos cuando se lleve a cabo un evento en específico.

4.3. Desarrollo

Al principio se buscó un simulador de la placa Rabbit pero ningún fabricante de Rabbit o terceras empresas a desarrollado uno. Rabbit brinda la posibilidad de debuggear sin estar conectado a la placa, pero sólo es posbile descubrir errores de sintáxis. Soporta los siguiente tipos de conexiones Ethernet:

- No LAN - se conecta la placa Rabbit al puerto Ethernet de la computadora mediante un RJ-45 cruzado.
- Micro - LAN - se conecta la placa y la computadora a un hub 10Base-T usando cables RJ-45 derechos.
- LAN - se conecta la placa a una red LAN preferentemente a la que está conectada la computadora (Se debe obtener una dirección IP).
- WAN - Se conecta la placa a Internet pero se recomienda programar y debuggear en la red local antes de conectar la placa a Internet.

  Utiliza una conexión 10/100Base-T a una velocidad de 10 Mbps. Cada RCM3700 tiene su propia MAC address. No soporta IPv6, solo IPv4. Si no se busca acceder al RCM3700 por Internet, es posible situarlo en la red interna utilizando una IP estática o bien por DHCP. Si se desea acceder a la placa por Internet, es posible situarla detrás del firewall configurando el firewall de forma tal que traduzca y mande paquetes de Internet al RCM3700. Se comenzaron a desarrollar diferentes pruebas con la finalidad de abarcar de a poco los alcances del proyecto. Las distintas programaciones correspondieron a la utilización de un protocolo

en particular, luego se intentó conjurar la utilización de los distintos protocolos en un archivo. Las distintas aplicaciones fueron compiladas en la memoria flash de la placa y ejecutadas en la RAM. El software utilizado para desarrollar las distintas pruebas fue el Dynamic C. Las páginas web empleadas para las distintas pruebas fueron diseñadas mediante la programación HTML.

4.4. Estrategia de programación

Se comenzó haciendo pruebas con los distintos puertos series de la placa, cuando se vio el claro manejo del puerto serie se buscó la forma de cargarle a la placa un pequeño servidor web. La programación del servidor web constituyó en la implementación del protocolo HTTP; una vez alcanzado el objetivo se buscó la forma de restringir el acceso al servidor web por medio de usuarios y contraseñas, por lo cual se implementó un método de autenticación dentro del mismo. En otro orden, considerando las amplias librerías que Rabbit ha desarrollado para el manejo del stack de directivas TCP/IP, se intentó implementar el envió de correos electrónicos por medio del protocolo SMTP. Como próximo objetivo se busca comandar un puerto serie de la placa por medio de un servidor web, ofreciendo distintos modos de operación, para lo cual se diseñó una página web específica utilizando el lenjuaje HTML - una imagen de la página web puede verse en el anexo -.

4.5. Forma de programación de la placa

Inicialmente, habiendo leído cierta literatura no oficial correspondiente a la placa Rabbit, se consideró que existía la forma de programar el RCM3700 por Ethernet. Visto y considerando que los manuales oficiales de Rabbit no mencionaban este tema se envió una consulta al Tech Support de Rabbit preguntando sobre esta prestación; la respuesta fue clara: no es posible programar la placa por Ethernet. La única forma de programación de la placa es mediante el puerto serie de la pc utilizando el cable de programación que viene incluído en el kit de desarrollo. El software utilizado para la programación es también Dynamic C.

4.6. Pruebas

Aqui se explica el funcionamineto de las distintas pruebas y programaciones que se realizaron. El código fuente de cada ejemplo se encuentra en el anexo correspondiente.

4.6.1. Switchcaracter.c

El programa transmite y recibe un string ASCII en el puerto C y E, y despliega en pantalla el mensaje recibido desde ambos puertos. Presionando los botones S1 y S2 se transmite el mensaje desde un puerto hacia el otro, presionando y soltando S1 se envía el mensaje desde el puerto C al E y presionando y soltando S2 se realiza lo inverso. La velocidad del puerto está seteada a 19200. En el prototyping board se debe conectar el TXC con el RXE y el RXC con el TXC.

4.6.2. Paridad.c

Permite hacer selecciones sobre el control de errores por paridad enviando repetidamente bytes con valores 0 - 127 desde el puerto D hacia el puerto C. Permite seleccionar entre generar paridad o no generarla en el puerto D. El puerto C siempre chequea la paridad. En el prototyping board se debe conectar el TXD al RXC. La velocidad del puerto está seteada a 9600 baudios.

4.6.3. Controlflujo.c

Demuestra el control de flujo por hardware del puerto serial, enviando un patrón de caracteres '*' desde el puerto D de la placa, TXD (PC0), a una velocidad de 115200 baudios. Un caracter a la vez es recibido desde el puerto D y es desplegado. En este caso RXC (PC3) se configura como el CTS input, detectando una señal de Clear to Send y TXC (PC2) es configurado como RTS output, señalando un estado de Ready to Send. En el prototyping board se jumpea el RXC con el TXC y el RXD con el TXD.

4.6.4. Autentificación.c

Permite que los usuarios se autentifiquen con un user y pass y se permite la visualización de una página web con una imagen y un mensaje. Se pueden configurar los usuarios que se deseen, por ahora solo está configurado para 3. Las opciones para la autentificación están implementadas en el código con un case. El main ejecuta un menú y según lo que el usuario presione establece las distintas opciones: habilitar/deshabilitar el usuario 1, habilitar/deshabilitar el usuario 2, habilitar/deshabilitar el usuario 3, autentificación básica, autentificación con resumen o sin autentificación.

4.6.5. Browsled.c

Permite controlar por medio de una página web el prendido y apagado de los leds DS1 y DS2 de la placa de desarrollo. Las opciones de prender/apagar cada uno de los botones se realiza en el main, primero se encuentra prendido

el led1 y apagado el led2 y luego se inicializa la placa con las configuraciones http correspondientes y se entra en loop dando opción a que se cambien el estado de los leds mediante la presión de los botones correspondientes en la página web

4.6.6. Mailnew.c

Permite detectar si algiuen presiono alguno de los dos botones que se encuentran el la placa de desarrollo, si es asi envia un mail avisando de lo ocurrido.

4.6.7. Browsnew.c

Permite autentificar usuarios (como lo hace el codigo autentificacion.c) y luego permite el prendido y apagado de los leds DS1 y DS2 de la placa de desarrollo (como la hace el codigo browsled.c)

Capítulo 5

Proximos Pasos

5.1. Introducción

Visto el avance alcanzado hasta el momento y habiendo completado ciertas etapas cabe realizar una propuesta detallada sobre los próximos pasos a tomar hasta la finalización del proyecto. En este capítulo se consideran nuevamente las distintas etapas y se establecen las distintas tareas a realizar.

5.2. Herramientas para la gestion de proyeco

- Se implementará el uso de una herramienta llamada UltraVNC la cual permite el manejo y la visualización remota de una pc conectada a la red externa configurando debidamente un router como gateway. El propósito es dejar conectada una pc las 24 hrs. del día a la placa Rabbit y permitir el desarrollo y las pruebas sobre la misma utilizando Internet y de esta forma lograr la programación remota de la placa.

5.3. Protocolos

- Se analizará detenidamente el modelo OSGI y se buscará corresponder el protocolo general para comandar los distintos módulos de salida con el mismo.

- Se investigará el protocolo RFI y su posible implementación tomando en cuenta las limitaciones de la placa en cuanto a velocidad.

- Se examinará el protocolo IrDA y su desarrollo en base a la confección de bases de datos.

- Se considerará el protocolo X10 y su posible implementación.

5.4. Implementación y programación

- Se programarán métodos de reseteo del sistema tanto en forma interna como externa.
- Se confeccionará una página web específica de configuración y se buscará su implementación dentro del sistema.
- Se estudiarán las posibles arquitecturas en base a la distribución de los procesos de hardware y software.
- Se comandrá el puerto serie por internet.
- Se buscará el correcto funcionamiento de cuatro puertos seriales comandados por el mismo bus.
- Se investigará la escalabilidad en base a los distintos protocolos a implementar.

5.5. Diseño

- Se elaborará el esquemático con el circuito correspondiente y sus componentes.
- Se determinará la alimentación del sistema y su debida protección en base a las necesidades del mismo.

Bibliografía

[Caprile, 2004.] Caprile, Sergio 2004.Introducción a Rabbit. Tutorial: CTU-004 [online]. Disponible en Internet: <http://www.cika.com/soporte/Tutorials/CTU-004_IntroRabbit.pdf>

[Casadomo] Casadomo. Pasarelas Residenciales [online]. Disponible en Internet:<http://www.casadomo.com/noticiasDetalle.aspx?c=49&m=15&idm=60&pat=14&n2=14>

[RS232C] RS232C. Comunicaciones RS232C [online]. Disponible en Internet:<http://eq3.uab.es/personal/baeza/comunicaciones/comunica.htm>

[RS232C] RS232C. Laboratorio RS232C [online]. Disponible en Internet: <http://www.sis.pitt.edu/ jarauz/docsus-fq/ene06/lab_rs232phy0105_10.doc>

[RS232] RS232. Estándar RS232 [online]. Disponible en Internet: <http://www.camiresearch.com/Data_Com_Basics/RS232_standard.html>

[RS232] RS232. Estándar RS232 [online]. Disponible en Internet: <http://www.euskalnet.net/shizuka/rs232.htm>

[Osgi] Osgi Alliance. - The Dynamic Module System for Java [online]. Disponible en Internet: <http://www.osgi.org >

[Osgi] Osgi Alliance. Artículo I sobre OSGI [online]. Disponible en Internet: <http://z0.saladeteletipos.com/twiki/pub/ProcesadorMultiprotocolar/BiblioProtocolos/Developing_Java_OSGi_applications_for_embedded_devices.pdf>

[Osgi] Osgi Alliance. Artículo II sobre OSGI [online]. Disponible en Internet: <http://z0.saladeteletipos.com/twiki/pub/ProcesadorMultiprotocolar/BiblioProtocolos/About_the_OSGi_Service_Platform.pdf>

[Osgi] Osgi Alliance. Especificación ofical de OSGI [online]. Disponible en Internet: <http://z0.saladeteletipos.com/twiki/pub/ProcesadorMultiprotocolar/BiblioProtocolos/OSGi_R4.core.pdf>

[HTTP] Protocolo HTTP. [online]. Disponible en Internet: <http://www.programacionweb.net/articulos/articulo/?num=409>

[HTTP] Protocolo HTTP. [online]. Disponible en Internet: <http://acs.barrapunto.org/articulos/trunk/LinuxActual/Apache/html/x49.html>

[HTTP] Protocolo HTTP. [online]. Disponible en Internet: <http://www.webopedia.com/TERM/H/HTTP.html>

[SMTP] Protocolo SMTP. [online]. Disponible en Internet: <http://www.ietf.org/rfc/rfc0821.txt>

[SMTP] Protocolo SMTP. [online]. Disponible en Internet: <http://www.programacionweb.net/articulos/articulo/?num=412>

[Rabbit] RCM3700 RabbitCore [online]. Disponible en Internet:<http://www.rabbitsemiconductor.com/products/rcm3700/>

[Rabbit] Ethernet Connection, Development Kit [online]. Disponible en Internet: <http://www.rabbitsemiconductor.com/products/EthernetConnKit/index.shtml>

[Rabbit] RCM3700 Manual de Usuario.

[Rabbit] Manual de usuario TCP/IP vol. 1.

[Rabbit] Manual de usuario TCP/IP vol. 2.

[Zytrax] RS232 Cables,Wiring and Pinouts [online]. Disponible en Internet: <http://www.zytrax.com/tech/layer_1/cables/tech_rs232.htm>

[Zytrax] Serial Interface Primer [online]. Disponible en Internet: <http://www.zytrax.com/tech/layer_1/cables/heavy.htm>

Parte II

Anexos

Apéndice A

Códigos Fuentes

A.1. Introducción

En esta sección se describen con diagramas de bloque las prestaciones de la placa RCM3700 y se documentan los distintos códigos utilizados para la programación de las distintas pruebas.

A.2. Código Switchcaracter.c

```
#class auto

#define DS1 6
#define DS2 7
#define S1  4
#define S2  7
#define ON  0
#define OFF 1

#define CINBUFSIZE    255
#define COUTBUFSIZE    255

#define EINBUFSIZE    255
#define EOUTBUFSIZE  255

nodebug
void msDelay(unsigned int delay)
{
   auto unsigned long done_time;

   done_time = MS_TIMER + delay;
   while( (long) (MS_TIMER - done_time) < 0 );
}


int pbRdSwitch(int swstate)
{
   if (swstate == S1)
      return (BitRdPortI(PFDR, swstate));
   else
      return (BitRdPortI(PBDR, swstate));
}

void pbWrLed(int led, int onoff)
{
   BitWrPortI(PFDR, &PFDRShadow, onoff, led);
}

main()
{
   auto int i, ch;
   auto char buffer[64];
   auto int sw1, sw2, led1, led2;
```

```
    static const char string1[] = {"Lo mando el puerto E al puerto C !!!\n\n\r"};
    static const char string2[] = {"Lo mando el puerto C al puerto E !!!\n\n\r"};

    brdInit();

    BitWrPortI(PEDR, &PEDRShadow, 0, 5);

    led1=led2=1;
    sw1=sw2=0;

     serEopen(19200);
    serEwrFlush();
     serErdFlush();

        serCopen(19200);
        serCwrFlush();
        serCrdFlush();

    memset(buffer, 0x00, sizeof(buffer));

    printf("\nArranco el programa\n\n\n\r");

    for(;;)
    {
        costate
        {
            if (pbRdSwitch(S1))
                abort;
            waitfor(DelayMs(50));
            if (pbRdSwitch(S1))
            {
                sw1=!sw1;
                abort;
            }
        }

        costate
        {
            if (pbRdSwitch(S2))
                abort;
            waitfor(DelayMs(50));
            if (pbRdSwitch(S2))
            {
                sw2=!sw2;
                abort;
            }
        }

        costate
        {
            if (sw1)
            {
                pbWrLed(DS1, ON);
                sw1=!sw1;

                memcpy(buffer, string2, strlen(string2));
                serCputs(buffer);
                memset(buffer, 0x00, sizeof(buffer));

                 i = 0;
                  while((ch = serEgetc()) != '\r')
                  {
                   if(ch != -1)
                   {
                      buffer[i++] = ch;
                   }
                }
                buffer[i++] = ch;
                  buffer[i]    = '\0';

                  printf("%s", buffer);

                memset(buffer, 0x00, sizeof(buffer));
                pbWrLed(DS1, OFF);
            }
        }

        costate
        {
            if (sw2)
            {
                pbWrLed(DS2, ON);
                sw2=!sw2;

                memcpy(buffer, string1, strlen(string1));
                  serEputs(buffer);
                  memset(buffer, 0x00, sizeof(buffer));
```

```
                i = 0;
                while((ch = serCgetc()) != '\r')
                {
                 if(ch != -1)
                 {
                    buffer[i++] = ch;
                 }
             }
                buffer[i++] = ch;
                buffer[i]   = '\0';

             printf("%s", buffer);
             pbWrLed(DS2, OFF);
          }
       }
    }
}
```

A.3. Código Paridad.c

```
#class auto

#define DINBUFSIZE 31
#define DOUTBUFSIZE 15
#define CINBUFSIZE 31
#define COUTBUFSIZE 31

const long baud_rate = 9600L;

main()
{
   auto char received;
   auto int i;
   auto int txconfig;

   brdInit();

   BitWrPortI(PEDR, &PEDRShadow, 0, 5);

   serDopen(baud_rate);
   serCopen(baud_rate);

   serDparity(PARAM_OPARITY);
   serCparity(PARAM_OPARITY);
   txconfig = PARAM_OPARITY;

   printf("Starting...\n");

   while (1)
   {
      costate
      {
         for (i = 0; i < 128; i++)
         {
            waitfor(DelayMs(10));
            waitfordone{ cof_serDputc(i); }
         }
           waitfor(serDwrFree() == DOUTBUFSIZE);
           waitfor(!((RdPortI(SDSR)&0x08) || (RdPortI(SDSR)&0x04)));
         yield;

         if (txconfig)
         {
            txconfig = PARAM_NOPARITY;
            printf("\nSin paridad\n");
         }
         else
         {
            txconfig = PARAM_OPARITY;
            printf("\nCon paridad\n");
         }
         serDparity(txconfig);
      }

      costate
      {
         waitfordone
         {
            received = cof_serCgetc();
         }
         printf("Se órecibi 0x%x\n", received);
         if (serCgetError() & SER_PARITY_ERROR)
         {
            printf("Error de paridad\n");
```

```
            }
         }
      }
}
```

A.4. Código Controlflujo.c

```
#class auto

#define DOUTBUFSIZE 31
#define DINBUFSIZE 15

#define SERD_CTS_PORT PCDR
#define SERD_CTS_BIT 3
#define SERD_RTS_PORT PCDR
#define SERD_RTS_SHADOW PCDRShadow
#define SERD_RTS_BIT 2

const long baud_rate = 115200L;

main()
{
   auto char send_buffer[128];
   auto char received;
   auto char fc_flag;
   auto int i;
   auto int j;

   brdInit();

   BitWrPortI(PEDR, &PEDRShadow, 0, 5);   //setear en estado bajo para habilitar rs232

   serDopen(baud_rate);

   printf("Comenzando...\n");

   serDflowcontrolOn();
   fc_flag = 1;
   printf("Control de flujo On\n");

   send_buffer[0] = 0;  // terminar en null
   for (i = 0;i < 8;i++)
   {
      for (j=0; j <= i; j++)
      {
         strcat(send_buffer, "*");
      }
      strcat(send_buffer, "\r\n");
   }

   while (1)
   {
      costate
      {
         for(i = 0; i < 3;i++)
         {
            waitfordone
            {
               cof_serDputs(send_buffer);
            }
         }
         if (fc_flag)
         {
            serDflowcontrolOff();
            fc_flag = 0;
            printf("Control de flujo Off\n");
         }
         else
         {
            serDflowcontrolOn();
            fc_flag = 1;
            printf("Control de flujo On\n");
         }
      }
      costate
      {
         waitfordone
         {
            received = cof_serDgetc();
         }
         putchar(received);
      }
   }
}
```

A.5. Código Autentificacion.c

```
/*****************************************************************************Ó
        AUTENTIFICACIN
*****************************************************************************/

#define TCPCONFIG     1  // Ethernet habilitado, PPP desabilitado,
                         // DHCP desabilitado, Runtime disable

#define HTTP_MAXBUFFER      512  // ñTamao del buffer HTTP (bytes) para agunatar
                                 // óinformacin recibida y transmitida

#define USE_HTTP_DIGEST_AUTHENTICATION 1

#define SSPEC_USERSPERRESOURCE    3  // Define que el únmero de usuarios es 3

#define HTTP_NO_FLASHSPEC  // Desabilitar el flashspec

#memmap xmem
#use "dcrtcp.lib" // óUtilizacin de la ílibrera dcrtcp.lib
#use "http.lib"   // óUtilizacin de la ílibrera http.lib

/*  ximport es una ófuncin de Dynamic C que toma la imagen binaria de un
    archivo, la ubica en la memoria extendida del controlador y asocia uní
    smbolo con la ódireccin ífsica de la imagen en el controlador */

#ximport "samples/tcpip/http/pages/static.html"    index_html
#ximport "samples/tcpip/http/pages/rabbit1.gif"  rabbit1_gif

/*  SSPEC_MIMETABLE retorna un mensaje indicando la respuesta a las
    peticiones entrantes al servidor HTTP mediante la ócomparacin de laó
    extensin de la ópeticin entrante coné sta lista y devuelve el segundo
    campo con el contenido */

SSPEC_MIMETABLE_START
   SSPEC_MIME(".html", "text/html"),
   SSPEC_MIME(".gif", "image/gif")
SSPEC_MIMETABLE_END

void main(void)
{
   int user1;
   int user2;
   int user3;
   int user1_enabled;
   int user2_enabled;
   int user3_enabled;
   int page1;
   int page2;
   int ch;

   printf("Presionar '1', '2', o '3' para deshabilitar/habilitar los 3 usuarios .\n");
   printf("Presionar 'b', 'r', o 's' para setear la óautentificacin como ásbica, resumida
        o sin autentificar .\n\n");

   sock_init();  // Inicializar el TCP/IP stack
   http_init();  // Inicializar el servidor web

   http_setauthentication(HTTP_DIGEST_AUTH);
   printf("Usando óautentificacin con resumen\n");

/* óDefinicin de usuarios y ñcontraseas */

   user1_enabled = 1;
   user2_enabled = 1;
   user3_enabled = 1;
   user1 = sauth_adduser("usuario1", "1", SERVER_HTTP);
   user2 = sauth_adduser("usuario2", "2", SERVER_HTTP);
   user3 = sauth_adduser("usuario3", "3", SERVER_HTTP);

   page1 = sspec_addxmemfile("/", index_html, SERVER_HTTP);
   sspec_adduser(page1, user1);
   sspec_adduser(page1, user2);
   sspec_adduser(page1, user3);
   sspec_setrealm(page1, "Admin");

   page2 = sspec_addxmemfile("index.html", index_html, SERVER_HTTP);
   sspec_adduser(page2, user1);
   sspec_adduser(page2, user2);
   sspec_adduser(page2, user3);
   sspec_setrealm(page2, "Admin");

   sspec_addxmemfile("rabbit1.gif", rabbit1_gif, SERVER_HTTP);

/* tcp_reserveport le exige al servidor ignorar peticiones cuando no existe
   un socket disponible en el puerto correspondiente; guarda memoria pero puede
   causar delays cuando se actualiza una ápgina. */
```

```
    tcp_reserveport(80);

    while (1) {     // Fijarse que óbotn apreta el usuario

        if (kbhit()) {
            ch = getchar();
            switch (ch) {
            case '1':
                user1_enabled = !user1_enabled;
                if (user1_enabled) {
                    sspec_adduser(page1, user1);
                    sspec_adduser(page2, user1);
                    printf("Usuario 1 habilitado\n");
                }
                else {
                    sspec_removeuser(page1, user1);
                    sspec_removeuser(page2, user1);
                    printf("Usuario 1 deshabilitado\n");
                }
                break;

            case '2':
                user2_enabled = !user2_enabled;
                if (user2_enabled) {
                    sspec_adduser(page1, user2);
                    sspec_adduser(page2, user2);
                    printf("Usuario 2 habilitado\n");
                }
                else {
                    sspec_removeuser(page1, user2);
                    sspec_removeuser(page2, user2);
                    printf("Usuario 2 deshabilitado\n");
                }
                break;

            case '3':
                user3_enabled = !user3_enabled;
                if (user3_enabled) {
                    sspec_adduser(page1, user3);
                    sspec_adduser(page2, user3);
                    printf("Usuario 3 habilitado\n");
                }
                else {
                    sspec_removeuser(page1, user3);
                    sspec_removeuser(page2, user3);
                    printf("Usuario 3 deshabilitado\n");
                }
                break;

            case 'b':
                http_setauthentication(HTTP_BASIC_AUTH);
                printf("Usando óautentificacin ábsica\n");
                break;

            case 'r':
                http_setauthentication(HTTP_DIGEST_AUTH);
                printf("Usando óautentificacin con resumen\n");
                break;

            case 's':
                http_setauthentication(HTTP_NO_AUTH);
                printf("No usando óautentificacin\n");
                break;
            }
        }

        http_handler();    // debe llamarse el http_handler().
    }
}
```

A.6. Código Browseled.c

```
/***********************************************************
    browseled.c
***********************************************************/
#class auto

#define DS1 0x40   //led, port F bit 6 bitmask
#define DS2 0x80   //led, port F bit 7 bitmask


#define TCPCONFIG 1


```

```
#define TCP_BUF_SIZE 2048


#define HTTP_MAXSERVERS 2
#define MAX_TCP_SOCKET_BUFFERS 2

#define REDIRECTHOST     _PRIMARY_STATIC_IP


/*
 *  REDIRECTTO is used by each ledxtoggle cgi's to tell the
 *  browser which page to hit next.  The default REDIRECTTO
 *  assumes that you are serving a page that does not have
 *  any address translation applied to it.
 *
 */

#define REDIRECTTO       "http://" REDIRECTHOST "/index.shtml"

#memmap xmem
#use "dcrtcp.lib"
#use "http.lib"

/*
 *  Notice that we have ximported in the source code for
 *  this program.  This allows us to <!--#include file="ssi.c"-->
 *  in the pages/showsrc.shtml.
 *
 */

#ximport "samples/rcm3700/tcpip/pages/browseled.shtml"   index_html
#ximport "samples/rcm3700/tcpip/pages/rabbit1.gif"       rabbit1_gif
#ximport "samples/rcm3700/tcpip/pages/ledon.gif"         ledon_gif
#ximport "samples/rcm3700/tcpip/pages/ledoff.gif"        ledoff_gif
#ximport "samples/rcm3700/tcpip/pages/button.gif"        button_gif
#ximport "samples/rcm3700/tcpip/pages/showsrc.shtml"     showsrc_shtml
#ximport "samples/rcm3700/tcpip/browseled.c"             browseled_c

/*
 *  In this case the .html is not the first type in the
 *  type table.  This causes the default (no extension)
 *  to assume the shtml_handler.
 *
 */

/* the default for / must be first */
SSPEC_MIMETABLE_START
   SSPEC_MIME_FUNC(".shtml", "text/html", shtml_handler),
   SSPEC_MIME(".html", "text/html"),
   SSPEC_MIME(".gif", "image/gif"),
   SSPEC_MIME(".cgi", "")
SSPEC_MIMETABLE_END

/*
 *  Each ledx contains a text string that is either
 *  "ledon.gif" or "ledoff.gif"  This string is toggled
 *  each time the ledxtoggle.cgi is requested from the
 *  browser.
 *
 */

char led1[15];
char led2[15];

/*
 *  Instead of sending other text back from the cgi's
 *  we have decided to redirect them to the original page.
 *  the cgi_redirectto forms a header which will redirect
 *  the browser back to the main page.
 *
 */

int led1toggle(HttpState* state)
{
   if (strcmp(led1,"ledon.gif")==0)
      strcpy(led1,"ledoff.gif");
   else
      strcpy(led1,"ledon.gif");

   cgi_redirectto(state,REDIRECTTO);
   return 0;
}

int led2toggle(HttpState* state)
{
   if (strcmp(led2,"ledon.gif")==0)
      strcpy(led2,"ledoff.gif");
   else
```

```
        strcpy(led2,"ledon.gif");

    cgi_redirectto(state,REDIRECTTO);
    return 0;
}


SSPEC_RESOURCETABLE_START
    SSPEC_RESOURCE_XMEMFILE("/", index_html),
    SSPEC_RESOURCE_XMEMFILE("/index.shtml", index_html),
    SSPEC_RESOURCE_XMEMFILE("/showsrc.shtml", showsrc_shtml),
    SSPEC_RESOURCE_XMEMFILE("/rabbit1.gif", rabbit1_gif),
    SSPEC_RESOURCE_XMEMFILE("/ledon.gif", ledon_gif),
    SSPEC_RESOURCE_XMEMFILE("/ledoff.gif", ledoff_gif),
    SSPEC_RESOURCE_XMEMFILE("/button.gif", button_gif),
    SSPEC_RESOURCE_XMEMFILE("browseled.c", browseled_c),
    SSPEC_RESOURCE_ROOTVAR("led1", led1, PTR16, "%s"),
    SSPEC_RESOURCE_ROOTVAR("led2", led2, PTR16, "%s"),
    SSPEC_RESOURCE_FUNCTION("/led1tog.cgi", led1toggle),
    SSPEC_RESOURCE_FUNCTION("/led2tog.cgi", led2toggle),
SSPEC_RESOURCETABLE_END


void update_outputs()
{
    auto int value;

    value=PFDRShadow&0x3F;    //on state for leds

    /* update O0 */
    if (strcmp(led1,"ledon.gif"))
        value|=DS1;

    /* update O1 */
    if (strcmp(led2,"ledon.gif"))
        value|=DS2;

    WrPortI(PFDR, &PFDRShadow, value);
}

main()
{

    brdInit();             //initialize board for this demo

    strcpy(led1,"ledon.gif");
    strcpy(led2,"ledoff.gif");

    sock_init();
    http_init();
    tcp_reserveport(80);

    while (1)
    {
        update_outputs();
        http_handler();

    }
}

#nodebug
```

A.7. Código Mailnew.c

```
/* DC9 update
 Esto es requerido por Dynamic C óversin 9, para utilizar
una IP fija en programa y la interfaz Ethernet por defecto.
No es necesario en DC8, se autodefinen en las libs */
#define TCPCONFIG 1
#define USE_ETHERNET     1
/* Fin DC9 update */


#define FROM      "ac@netgate.com.uy"//"scaprile@ellie"
#define TO        "rossanam@gmail.com"//"scaprile@ellie"
#define SUBJECT   "Rabbit email"
#define BODY1     "Alguien ópresion el óbotn"
#define BODY2     "\r\nSaludos, RCM3700."

#define SMTP_SERVER "smtp.netgate.com.uy" //"192.168.1.50"

#define MY_IP_ADDRESS    "192.168.1.102" //"192.168.1.54"
#define MY_NETMASK       "255.255.255.0"

#define MY_GATEWAY       "192.168.1.1"
```

```
#define MY_NAMESERVER   "192.168.1.1" //"200.42.0.108"

#define SMTP_DEBUG

#memmap xmem
#use "dcrtcp.lib"
#use "smtp.lib"


main()
{
int i,mail;
char body[256];

    sock_init();
    mail=0;

   // WrPortI(PFFR,&PEFRShadow,0);
    WrPortI(PBDDR,&PBDDRShadow,0x7F);
   //   WrPortI(PFCR,&PECRShadow,0);
   //   WrPortI(PFDR,&PEDRShadow,0xc0);              /* apaga ambos LEDs */

    while(1){
            if(!BitRdPortI(PFDR,4)) {
                sprintf(body,"%s S1 %s",BODY1,BODY2);
                mail=1;
            }
            if(!BitRdPortI(PBDR,7)) {
                sprintf(body,"%s S2 %s",BODY1,BODY2);
                mail=1;
            }
            if(mail){
                printf("Listo a enviar:\n\t%s\n",body);
                smtp_sendmail(TO, FROM, SUBJECT, body);
                printf("Enviando...\n");
                while(smtp_mailtick()==SMTP_PENDING);
                printf("Listo.\n");
                mail=0;
                for(i=0;i<30000;i++);
            }
    }
}
```

A.8. Código Browsnew.c

```

#class auto

#define DS1 0x40  //led, port F bit 6 bitmask
#define DS2 0x80  //led, port F bit 7 bitmask

/**********************************
 * Seccion de óconfiguracin        *
 *--------------------------------*
 * Todos los campos de esta seccion*
 * debes ser altrados de acuerdo   *
 * la configuracion de la red local*
 **********************************/
/*
 * Mirar LIB\TCPIP\TCP_CONFIG.LIB para obtener instruciones
 * de como seterar la configuracion.
 */
#define TCPCONFIG 1

/*
 * TCP/IP modification - reduce TCP socket buffer
 * size, to allow more connections. This can be increased,
 * with increased performance, if the number of sockets
 * are reduced.  Note that this buffer size is split in
 * two for TCP sockets--1024 bytes for send and 1024 bytes
 * for receive.
 */
#define TCP_BUF_SIZE 2048


#define USE_HTTP_DIGEST_AUTHENTICATION 1
#define SSPEC_USERSPERRESOURCE   3

/*
 * Configuracion del servidor Web
 */

/*
 * Define el numero de HTTP servers y los socket buffers.
 * Con tcp_reserveport(), menos los servidores HTTP servers que sean necesarios.
```

```
 */
#define HTTP_MAXSERVERS 2
#define MAX_TCP_SOCKET_BUFFERS 2
#define HTTP_MAXBUFFER      512


/*
 * Our web server as seen from the clients.
 * This should be the address that the clients (netscape/IE)
 * use to access your server. Usually, this is your IP address.
 * If you are behind a firewall, though, it might be a port on
 * the proxy, that will be forwarded to the Rabbit board. The
 * commented out line is an example of such a situation.
 */
#define REDIRECTHOST      _PRIMARY_STATIC_IP
// #define REDIRECTHOST "my.host.com:8080"

/************************************
 * Fin de la seccion de configuration*
 ************************************/

/*
 *  REDIRECTTO is used by each ledxtoggle cgi's to tell the
 *  browser which page to hit next.  The default REDIRECTTO
 *  assumes that you are serving a page that does not have
 *  any address translation applied to it.
 *
 */

#define REDIRECTTO       "http://" REDIRECTHOST "/index.shtml"

#memmap xmem
#use "dcrtcp.lib"
#use "http.lib"

/*
 *  Notice that we have ximported in the source code for
 *  this program.  This allows us to <!--#include file="ssi.c"-->
 *  in the pages/showsrc.shtml.
 *
 */

#ximport "samples/rcm3700/tcpip/pages/browseled.shtml"   index_html
#ximport "samples/rcm3700/tcpip/pages/rabbit1.gif"       rabbit1_gif
#ximport "samples/rcm3700/tcpip/pages/ledon.gif"         ledon_gif
#ximport "samples/rcm3700/tcpip/pages/ledoff.gif"        ledoff_gif
#ximport "samples/rcm3700/tcpip/pages/button.gif"        button_gif
#ximport "samples/rcm3700/tcpip/pages/showsrc.shtml"     showsrc_shtml
#ximport "samples/rcm3700/tcpip/browseled.c"             browseled_c

#ximport "samples/tcpip/http/pages/static.html"    index1_html


/*
 *  In this case the .html is not the first type in the
 *  type table.  This causes the default (no extension)
 *  to assume the shtml_handler.
 *
 */

/* the default for / must be first */
SSPEC_MIMETABLE_START
   SSPEC_MIME_FUNC(".shtml", "text/html", shtml_handler),
   SSPEC_MIME(".html", "text/html"),
   SSPEC_MIME(".gif", "image/gif"),
   SSPEC_MIME(".cgi", "")
SSPEC_MIMETABLE_END

/*
 *  Each ledx contains a text string that is either
 *  "ledon.gif" or "ledoff.gif"  This string is toggled
 *  each time the ledxtoggle.cgi is requested from the
 *  browser.
 *
 */

char led1[15];
char led2[15];

/*
 *  Instead of sending other text back from the cgi's
 *  we have decided to redirect them to the original page.
 *  the cgi_redirectto forms a header which will redirect
 *  the browser back to the main page.
 *
 */

int led1toggle(HttpState* state)
```

```
{
   if (strcmp(led1,"ledon.gif")==0)
      strcpy(led1,"ledoff.gif");
   else
      strcpy(led1,"ledon.gif");

   cgi_redirectto(state,REDIRECTTO);
   return 0;
}

int led2toggle(HttpState* state)
{
   if (strcmp(led2,"ledon.gif")==0)
      strcpy(led2,"ledoff.gif");
   else
      strcpy(led2,"ledon.gif");

   cgi_redirectto(state,REDIRECTTO);
   return 0;
}


SSPEC_RESOURCETABLE_START
   SSPEC_RESOURCE_XMEMFILE("/", index_html),
   SSPEC_RESOURCE_XMEMFILE("/index.shtml", index_html),
   SSPEC_RESOURCE_XMEMFILE("/showsrc.shtml", showsrc_shtml),
   SSPEC_RESOURCE_XMEMFILE("/rabbit1.gif", rabbit1_gif),
   SSPEC_RESOURCE_XMEMFILE("/ledon.gif", ledon_gif),
   SSPEC_RESOURCE_XMEMFILE("/ledoff.gif", ledoff_gif),
   SSPEC_RESOURCE_XMEMFILE("/button.gif", button_gif),
   SSPEC_RESOURCE_XMEMFILE("browseled.c", browseled_c),
   SSPEC_RESOURCE_ROOTVAR("led1", led1, PTR16, "%s"),
   SSPEC_RESOURCE_ROOTVAR("led2", led2, PTR16, "%s"),
   SSPEC_RESOURCE_FUNCTION("/led1tog.cgi", led1toggle),
   SSPEC_RESOURCE_FUNCTION("/led2tog.cgi", led2toggle),
SSPEC_RESOURCETABLE_END


void update_outputs()
{
   auto int value;

   value=PFDRShadow&0x3F;   //on state for leds

   /* update O0 */
   if (strcmp(led1,"ledon.gif"))
      value|=DS1;

   /* update O1 */
   if (strcmp(led2,"ledon.gif"))
      value|=DS2;

   WrPortI(PFDR, &PFDRShadow, value);
}

main()
{
   int user1;          //
   int user2;       //
   int user3;          //
   int user1_enabled;   //
   int user2_enabled;    //
   int user3_enabled;    //
   int page1;          //
   int page2;         //
   int ch;            //

   printf("Presionar '1', '2', o '3' para deshabilitar/habilitar los 3 usuarios.\n"); //
   printf("Presionar 'b', 'r', o 's' para setear la óautentificacin como ábsica, resumida o sin autentificar.\n\n");   //

   strcpy(led1,"ledon.gif");
   strcpy(led2,"ledoff.gif");

   brdInit();           //initialize board for this demo
   sock_init();
   http_init();
   tcp_reserveport(80);

   http_setauthentication(HTTP_DIGEST_AUTH); //
   printf("Usando óautentificacin con resumen\n");//
   /* óDefinicin de usuarios y ñcontraseas */
   user1_enabled = 1; //
   user2_enabled = 1;  //
   user3_enabled = 1;   //
   user1 = sauth_adduser("usuario1", "1", SERVER_HTTP);//
   user2 = sauth_adduser("usuario2", "2", SERVER_HTTP);  //
```

```
    user3 = sauth_adduser("usuario3", "3", SERVER_HTTP);   //

    page1 = sspec_addxmemfile("/", index_html, SERVER_HTTP); //
    sspec_adduser(page1, user1);                              //
    sspec_adduser(page1, user2);                              //
    sspec_adduser(page1, user3);                              //
    sspec_setrealm(page1, "Admin");                           //

    page2 = sspec_addxmemfile("index.html", index_html, SERVER_HTTP); //
    sspec_adduser(page2, user1);                                       //
    sspec_adduser(page2, user2);                                       //
    sspec_adduser(page2, user3);                                       //
    sspec_setrealm(page2, "Admin");                                    //

    sspec_addxmemfile("rabbit1.gif", rabbit1_gif, SERVER_HTTP);      //


    while (1)
    {
    update_outputs();
    http_handler();

    }
}

#nodebug
```

Apéndice B

Imagenes paginas Web

B.1. Manejo del puerto serie por Internet

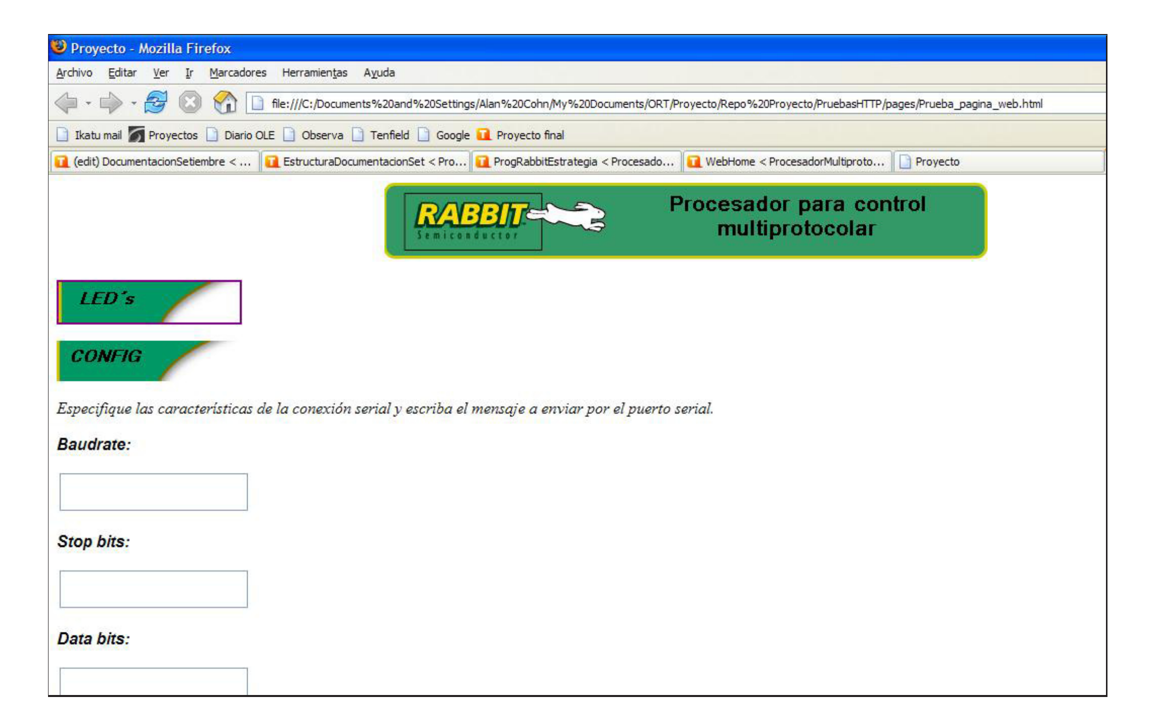


Figura B.1: Página Web

Apéndice C

Licencias

C.1. Licencia BSD

Copyright 1994-2006 The FreeBSD Project. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. THIS SOFTWARE IS PROVIDED BY THE FREEBSD PROJECT "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE FREEBSD PROJECT OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The views and conclusions contained in the software and documentation are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the FreeBSD Project.

Fuente: http://www.freebsd.org/copyright/freebsd-license.html

C.2. Licencia GNU — General Public Licence (GPL)

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU General Public License

Terms and Conditions For Copying, Distribution and Modification

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

    a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original

licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will

be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and ”any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

No Warranty

11. Because the program is licensed free of charge, there is no warranty for the program, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holders and/or other parties provide the program ”as is” without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the program is with you. Should the program prove defective, you assume the cost of all necessary servicing, repair or correction.

12. In no event unless required by applicable law or agreed to in writing will any copyright holder, or any other party who may modify and/or redistribute the program as permitted above, be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the program (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the program to operate with any other programs), even if such holder or other party has been advised of the possibility of such damages.

End of Terms and Conditions

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the ”copyright” line and a pointer to where the full notice is found.

> ¡one line to give the program’s name and a brief idea of what it does.¿
> Copyright (C) ¡year¿¡name of author¿
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) ¡year¿¡name of author¿
> Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
> This is free software, and you are welcome to redistribute it under certain conditions; type ’show c’ for details.

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w` and `show c`; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a ”copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program
> ’Gnomovision’ (which makes passes at compilers) written by James Hacker.
>
> ¡signature of Ty Coon¿, 1 April 1989
> Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Fuente: http://www.gnu.org/copyleft/gpl.html

- Traducción no oficial al español: http://gugs.sindominio.net/licencias/gples.html